Behavioral Self-organization in Lifelike Agents

Jiming Liu **Hong Qin**

Department of Computing Studies
Hong Kong Baptist University
Kowloon Tong, Hong Kong

Abstract

This paper is concerned with the acquisition of behaviors in lifelike synthetic agents that interact with virtual graphical environments. The goal of the agent behavioral acquisition is to select an effective behavioral pattern that adapts to the conditions of the agent environment, and in addition, to determine the corresponding behavioral parameters. In doing so, the lifelike agents will employ a dual-level behavioral self-organizing (BSo) approach, in which the high-level acquires a conditioned association from the presently sensed state of the environment to the requirement of a desired motion as well as a plausible behavioral pattern to enable such a motion, whereas the low-level computes the optimal parameters for the identified behavior in fulfilling the motion requirement.

Keywords: Behavioral Self-organization (BSo), lifelike characters, synthetic agents.

1 Introduction

In computer animation, lifelike agents are essential in producing the believable effects of living characters in response to uncontrolled, dynamically unfolding situations. Good examples of such agents are the ALIVE system as developed by Maes *et al* [14] and the Artificial Fishes by Terzopoulos *et al* [16]. The lifelike agents may constantly monitor the changes in their environments and proactively exhibit certain behaviors with a purpose. At the same time, the agents may also be directed or controlled through several levels of communication [5]. The work presented here explores a computational approach for constructing the underlying mechanism to allow for the behavioral acquisition in the lifelike agents. This approach, while drawing on Kohonen's self-organizing feature map principles, advocates a dual-level learning process that involves a high-level behavioral pattern identification and a low-level parameterization for realising the identified behavior. In order to limit our scope, here we shall not focus on the physical dynamics of the agents once a behavior is selected.

Autonomous Agents 98 Minneapolis MN USA

1.1 Previous Work

There have been several studies tackling the problem of articulated figure generation. Some of them utilized inverse Lagrangian dynamics algorithms to compute inertial motions based on carefully studied gait determinants [11, 17], whereas others applied the partial dynamics of some specific parts of the body such as legs and arms in order to reduce the computational complexity involved in the motion generation [1]. These studies have, to a certain extent, shared one thing in common; namely, the realistic motion was achieved by solving either complete or partial kinematic and dynamic equations. Two questions that remain are (1) how a believable movement can be most *efficiently* generated, and (2) how a lifelike agent can select an appropriate behavior in response to not only the given goal but also some *unpredictable* conditions in its environment [13]. This issue is particularly relevant if we are to develop synthetic agents that can "survive", autonomously, and acquire previously undefined behaviors.

Related to our work are some of the previous studies on behavior selection and emergence. For example, Maes [12] developed a selection mechanism that emerged an action by spreading activation energy over a behavior network. In relation to search based behavior selection, Liu *et al* [10] proposed and implemented an evolutionary strategy based method that enabled an animated creature to acquire its adaptive behaviors in a 3D graphical environment. Auslander *et al* [2] developed a system that contained banked stimulus-response controllers dynamically selected through an optimization algorithm. Ventrella [18] studied the possibility of emerging the structure and locomotion behaviors of an animate using genetic algorithms [7, 8], his system used a model of specifically tailored qualitative forward dynamics to generate gravitational, inertial, momentum, frictional, and dampening effects. Sims [15] developed a system in which both animated 3D creature bodies (i.e., morphology) and their neural control systems (i.e., virtual brains) were genetically evolved. Finally, another important aspect in autonomous agents should be mentioned, namely, the emotion of an agent. A number of researchers have already addressed this issue [3, 6].

In this paper, we describe a self-organizing feature map based learning approach to behavioral acquisition. The originality of our work lies in that we directly address the issues of behavioral learning from the point of view of concurrent behavioral conditioning through two interrelated levels of organization, one for patterns and another for parameterization.

1.2 Problem Statement

Here we shall specifically consider the following problem: Given a synthetic graphical agent that is equipped with (1) several sensors for identifying the current state of its virtual environment, in which the agent is in, and the internal state of the agent, and (2) a set of primitive motion behavioral patterns, how to enable the agent to gradually acquire its reactive behavior from the virtual environment, while attaining a certain goal. An example of the goal for the agent would be to pass over, in a certain direction, several objects as encountered from its environment.

1.3 Organization of the Paper

The remainder of this paper is organized as follows: Section 2 provides the details on our self-organization based dual-level behavioral acquisition approach, including both the computational framework and the learning algorithm. In order to further demonstrate the developed behavioral acquisition approach, Section 3 presents an implemented lifelike agent, called Athlete, with a description of its structure, sensory input, primitive behavioral patterns, and learning in a virtual environment. Finally, Section 4 concludes the paper by pointing out the technical contributions of our work as well as the potential for practical applications.

2 Lifelike Behavioral Self-organization (BSo)

In this section, we present the proposed behavioral self-organization (BSo) approach. In doing so, we first give an overview of the approach as well as an implemented system incorporating such an approach, this is followed by a more detailed description of the underlying algorithm to be applied.

2.1 An Overview

In our present work, we develop a dual-level learning approach to behavioral acquisition that involves the construction of a high-level behavioral pattern map and a low-level behavioral parameterization map. Both maps are created by applying Kohonen's self-organizing map updating rules [4, 9].

Specifically, at the high-level, the self-organizing map, denoted by H_SOM, provides the following mapping:

$$\text{H\_SOM: } \{S_{ext}, S_{int}, V_0\} \rightarrow \{\langle \Delta x_d, \Delta y_d \rangle, B_i\} \qquad (1)$$

where S_{ext} denotes the sensed external state of the agent environment. S_{int} denotes the sensed internal state of the agent itself. V_0 denotes an initial velocity of the agent. $\langle \Delta x_d, \Delta y_d \rangle$ denotes the desired positional change for the agent. B_i denotes a plausible behavioral pattern.

At the same time, the low-level consists of a collection of maps, denoted by L_SOMs, each of which defines the following mapping:

$$\text{L\_SOM: } \{\langle \Delta x_d, \Delta y_d \rangle, B_i, V_0\} \rightarrow \{\alpha\} \qquad (2)$$

where α denotes a behavioral parameter that characterizes behavioral pattern B_i, and hence determines the outcome of B_i, including the resulting displacement and velocity of the agent. In other words, a single L_SOM maps a desired movement $\langle \Delta x_a, \Delta y_a \rangle$ under V_0 onto a desirable behavioral parameter α.

The basic ideas behind the dual-level behavioral acquisition are as follows: At a certain time in the virtual environment, an agent selects from its H_SOM a desired motion requirement along with a plausible behavioral pattern to fulfill such a requirement. Next, the selected behavioral pattern will be used as an index to retrieve a corresponding low-level parameterization map, from which a desired behavioral parameter is determined, and thereafter the parameterized behavior is executed in the virtual environment. If the resulting motion of the agent deviates from the desired motion, the result will be recorded and used to update L_SOM. By the same token, H_SOM will also concurrently be updated using the obtained result of the actual motion by the agent. This process of dual-level behavioral self-organization continues as the agent is directed toward a goal location.

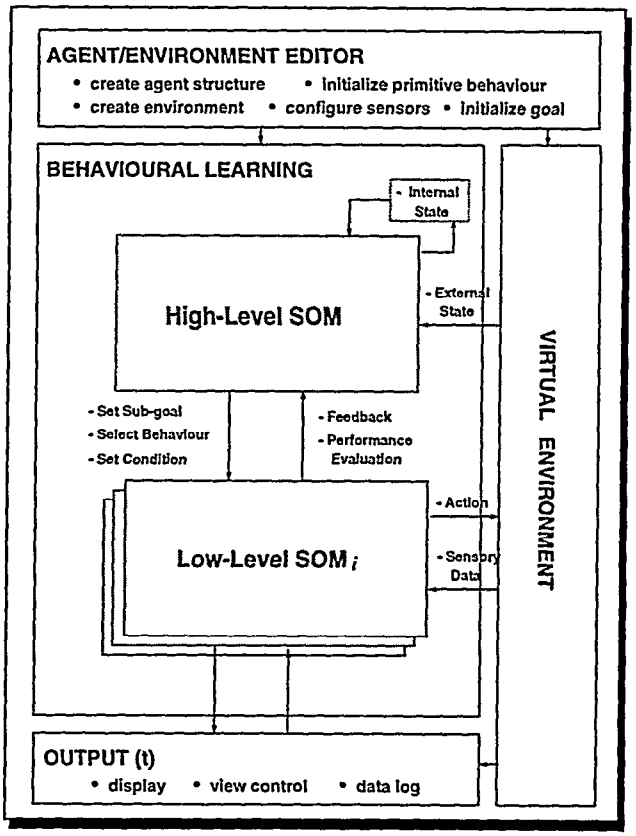

Figure 1: A schematic diagram of the system for building lifelike agents with a behavioral self-organizing (BSo) capability.

We have implemented the above mentioned behavioral acquisition scheme in a software system that allows for the construction of lifelike agents. Figure 1 provides a schematic diagram of the system in which the shaded regions depict the two interrelated self-organizing maps.

2.2 The Algorithm

The algorithm for implementing our proposed dual-level behavioral self-organization in a lifelike agent is given below. In the algorithm, the learning at both levels is performed during several trials. In other words, if the motion of the agent has resulted in collisions with the virtual environment for a number of times, a new motion requirement as well as a new behavioral pattern, in relation to the present states of the agent, will be selected.

```
initialize H_SOM and L_SOM_i
begin H_SOM learning
  select (Δx_d, Δy_d) and B_i from H_SOM
  counter1 ← 0
  while counter1 < 50 do
  begin L_SOM learning
    select L_SOM_i corresponding to output of H_SOM, B_i
    counter2 ← 0
    while counter2 < 20 do
    get α from L_SOM_i based on (Δx_d, Δy_d)
    execute behavior B_i in virtual environment
    if B_i does not cause collision then
      get (Δx_a, Δy_a)
      if sqrt((Δx_a − Δx_d)^2 + (Δy_a − Δy_d)^2) < 0.01 then
      break /*return success to H_SOM learning */
      else
        call update_L_SOM((Δx_a, Δy_a), B_i, V_0, α)
        call update_H_SOM((Δx_a, Δy_a), B_i, V_0)
      endif
    endif
    counter2 ← counter2 + 1
    endwhile
  end L_SOM learning
  if [L_SOM learning process return success] then
    call update_H_SOM((Δx_d, Δy_d), B_i, V_0)
    record the velocity resulting from behavior B_i
    break
  else
    randomly select (Δx, Δy) and B_n in the neighborhood of (Δx_a, Δy_a) and B_i, respectively
    counter1 ← counter1 + 1
  endif
  endwhile
  if goal is reached then
  exit
  else
    backtrack to previous state for relearning
  endif
end H_SOM learning
```

In the above algorithm, both the high-level pattern mapping (H_SOM) and the low-level parameterization (L_SOM) rely on a self-organizing process. In our present implementation, we apply Kohonen's algorithm for self-organizing feature maps [4, 9]. The specific updating rules in our present case are given below:

```
begin update_H_SOM((Δx, Δy), B_i, V_0)
  search for k such that encoded internal and external states are close or equal to
  W_*k and (W_vk − V_0) is minimal, where W_*k and W_vk are the weights of neuron k
  update weights for neuron k
    W_{B_ID}^{t+1}k ← B_i
    W_xk^{t+1} ← Δx
    W_yk^{t+1} ← Δy
    W_vk^{t+1} ← V_0
  update weights for the neighboring neuron m of neuron k
    (i.e., the internal and external states of neuron m is the
    same as those of neuron k and W_{B_ID}m ≡ B_i)
    W_xm^{t+1} ← W_xm^t + γ_t G_t[DIST(k, m)](Δx − W_xm^t)
    W_ym^{t+1} ← W_ym^t + γ_t G_t[DIST(k, m)](Δy − W_ym^t)
    W_vm^{t+1} ← W_vm^t + γ_t G_t[DIST(k, m)](V_0 − W_vm^t)
  t ← t + 1
end update_H_SOM
```

In the above update_H_SOM procedure, t = counter1 is the number of times to update H_SOM. $W_{B_ID}k, W_{xk}, W_{yk}$, and W_{vm} are the weights of neuron k in H_SOM. Learning coefficient $\gamma_t = (0.95)^t$. Decreasing function $G_t(d) = e^{-d^2/2\theta_t^2}$. $\theta_t = (0.95)^t$. $DIST(k, m) = |W_{vm}^t - W_{vk}^t|$.

```
begin update_L_SOM((Δx_a, Δy_a), B_i, V_0, α)
  find s = arg min_r((w_xr − Δx_a)^2 + (w_yr − Δy_a)^2)
  where s, r are neuron indices, and w_xs, w_ys are the weights of neuron s
  update the weights for the neighboring neuron j of neuron s:
    w_xj^{r+1} ← w_xj^r + β_r g_r[dist(s, j)](Δx_a − w_xj^r)
    w_yj^{r+1} ← w_yj^r + β_r g_r[dist(s, j)](Δy_a − w_yj^r)
    w_αj^{r+1} ← w_αj^r + β_r g_r[dist(s, j)](α − w_αj^r)
  r ← r + 1
end update_L_SOM
```

In our present implementation, we allow all neurons to be connected with each other. τ = counter2. Learning coefficient $\beta_\tau = (0.9)^\tau$. Decreasing function $g_\tau(d) = e^{-d^2/2\delta_\tau^2}$. $\delta_\tau = \delta_0(0.9)^\tau$, $\delta_0 = 0.95$. $dist(i, j) = |w_{aj}^\tau - w_{ai}^\tau|$.

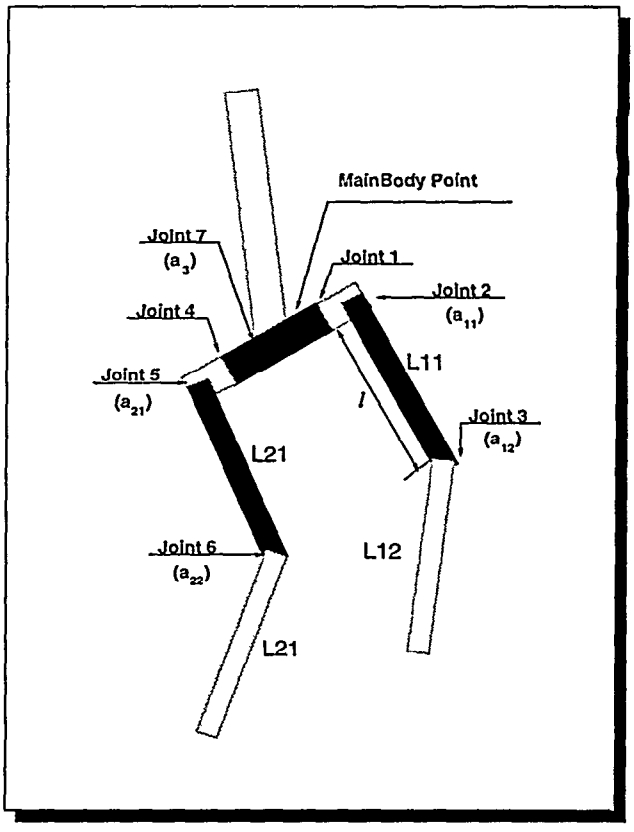

Figure 2: The kinematic structure of Athlete.

3 The Athlete Agent

In order to provide an example of behavioral self-organization, in what follows we present a lifelike agent, called Athlete, that interacts with its virtual environment and gradually acquires its reactive behavior during the process of attaining a global goal. Specifically speaking, the goal of Athlete is to go from one end of the environment to another end, while bypassing any graphical objects as encountered. Both the agent and its environment can be created using the system as described in Figure 1.

3.1 Athlete Structure

The kinematic structure of Athlete is composed of eight links, interconnected with seven revolute joints. The details on the arrangement of these components are illustrated in Figure 2. It should be mentioned that in our present experiments, only five out of the seven joints have been used to perform and update various reactive behaviors. These joints are a_{11}, a_{12}, a_{21}, and a_{22} at the two legs and a_3 at the main body (refer to the labels in the schematic diagram of Figure 2).

3.2 Virtual Athlete Environment

For the sake of illustration, we have created a graphical environment for Athlete to interact with, as shown in Figure 3. This environment contains several distinct graphical objects; namely,

- Flat_Track_Field areas 1, 3, 5, 7, and 9, separating other environment conditions,

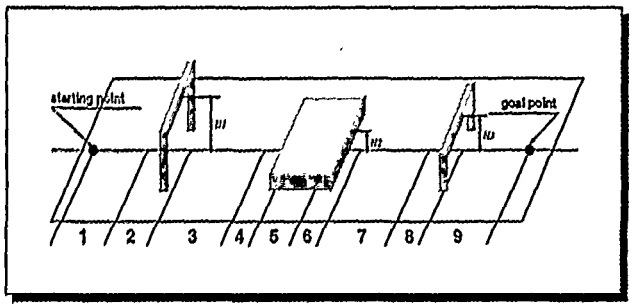

Figure 3: The virtual Athlete environment. The different areas are labeled from 1 to 9.

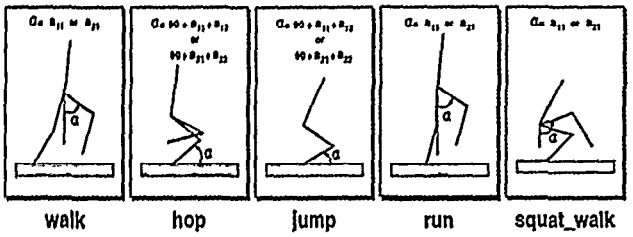

Figure 4: The definition of α parameters in five primitive behavioral patterns.

- High_Hurdle area 2,
- Up_Platform area 4,
- Down_Platform area 6, and
- Low_Hurdle area 8.

As also labeled in Figure 3, Athlete is required to go from an initial starting location to a goal location.

3.3 External and Internal States

There are three virtual sensors mounted to the main body of Athlete. These sensors would enable Athlete to extract a model of its local environment conditions, i.e., the presence of an object as well as the dimensions of the object such as H_1, H_2, and H_3 in Figure 3, and hence identify its current external state in the environment. In our present example, the detected external states correspond directly to the aforementioned five distinct environment conditions. In addition to the external states, Athlete can also evaluate its present internal state in terms of whether it is in a standing or squatty position. The external and internal state representations would serve as the necessary stimuli for Athlete to condition its reactive behaviors using the dual-level mechanism.

3.4 Primitive Behavioral Patterns

In reaction to its sensed external and internal states, Athlete will select one of the five predefined primitive behavioral patterns, namely, {walk, hop, jump, run, squat_walk}, in an attempt to achieve a certain desired motion. The actual motion outcome from each behavioral pattern, including both the displacement and the resulting velocity of the agent, is determined by the following three factors:

1. the relative spatial configuration and co-ordination of the two legs that define the behavioral pattern,

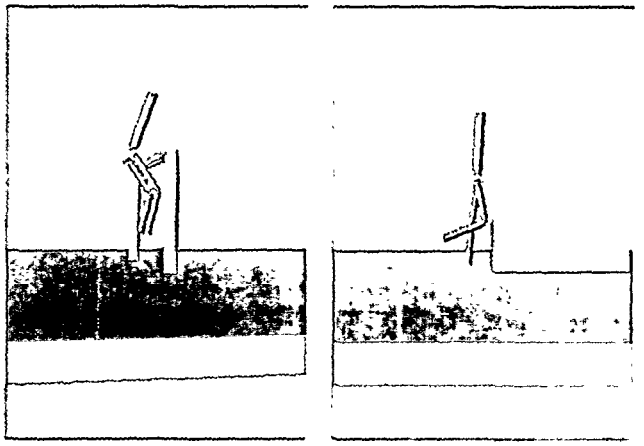

Figure 5: (a) An attempt to pass the High_Hurdle with jump has resulted in a collision. (b) One of the legs in a walk behavior has been found to be too close to the Up_Platform. This information will be incorporated into the dual-level behavioral self-organizing process.

2. an initial velocity, V_0, sensed before executing the behavioral pattern, and
3. the actual value of an α parameter specifically defined for the behavioral pattern.

Figure 4 illustrates the definition of the α parameter in each of the five Athlete behavioral patterns. The exact quantitative relationships among the motion outcome, the α parameter, and the initial velocity of the agent main body are given in the appendix.

3.5 Behavioral Self-organization Results

3.5.1 Dual-level Behavioral Acquisition Revisited

From a predefined initial location in its virtual environment, Athlete starts to move towards the final goal location. In doing so, the agent initializes a H_SOM and a collection of L_SOMs, and updates these behavioral conditioning maps based on the algorithm as given in Section 2.2. In other words, whenever the agent encounters (or feels) a distinct environment condition with the three mounted sensors, it generates a desired motion requirement and at the same time, selects a plausible behavioral pattern. The generated requirement along with the selected behavioral pattern is then passed to the low-level behavioral parameterization module in order to validate whether or not the desired motion can be achieved. If not, the agent self-organizes its maps with the actual motion outcome obtained. This cycle of learning repeats as the agent moves from one region to another.

Here it should be pointed out that there are two ways to terminate the validation at the low-level, namely, (1) the successful completion of a parameterized motion behavior and (2) the collision of Athlete link with some object(s) in the environment. Figures 5(a) and (b) present two unsuccessful attempts of the agent. As shown in Figure 5(a), the agent selected a jump behavior in Area 2 of the virtual environment, trying to pass over the High_Hurdle. However, since the maximal jumping height was lower than that of the hurdle as had observed during the low-level behavioral learning, such a reactive behavioral pattern would fade away from the high-level behavioral pattern conditioning map. In Figure 5(b), it is shown that the agent, in sensing the Up_Platform condition, continued to

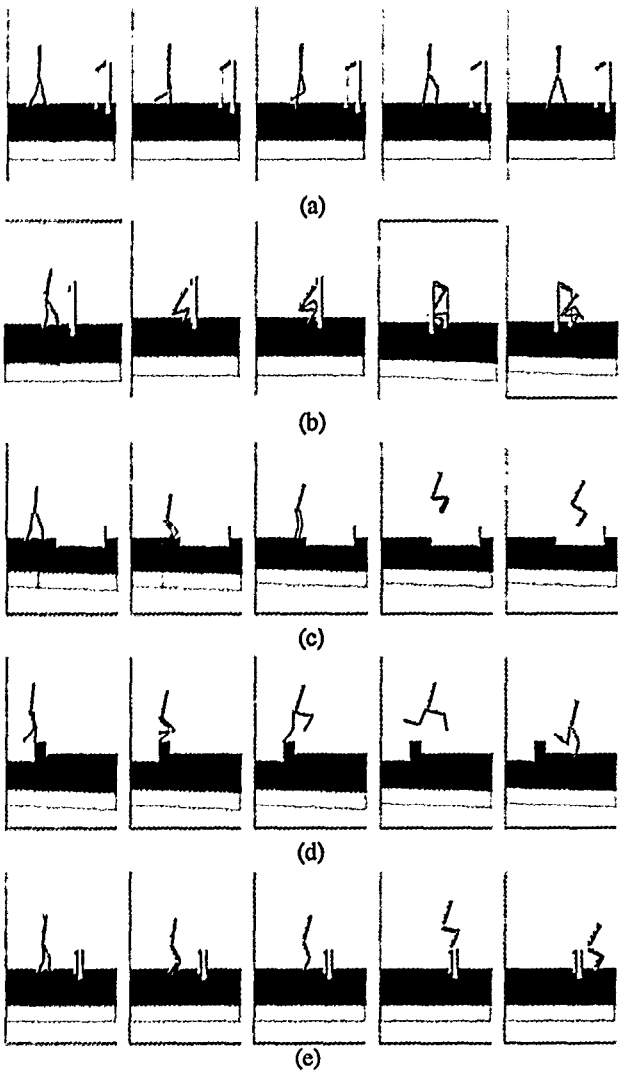

Figure 6: (a) Approaching, through walk, toward the High_Hurdle in Area 1 of the virtual environment. (b) Squat_walking beneath the High_Hurdle in Area 2. (c) Jumping up to the Up_Platform in Area 4. (d) Hopping down from the Down_Platform in Area 6. (e) Jumping over the Low_Hurdle in Area 8.

select a previously adopted behavior, walk, which also caused a collision with the environment as the stage was higher than what the walk behavior could reach.

As can readily noted based on the discussion in Section 3.4, failures in producing a suitable behavioral response may be caused by different factors; for instance, (1) correct motion requirement and behavioral parameterization but wrong pattern, (2) correct pattern and initial velocity but wrong α value, and/or (3) correct pattern and behavioral parameterization but wrong initial velocity. When Case (3) occurs, the agent will backtrack its behaviors to one of the earlier states. In this respect, the high-level behavioral pattern conditioning in our proposed dual-level acquisition scheme has incorporated a notion of interdependency among a sequence of reactive behaviors (i.e., a composite behavior).

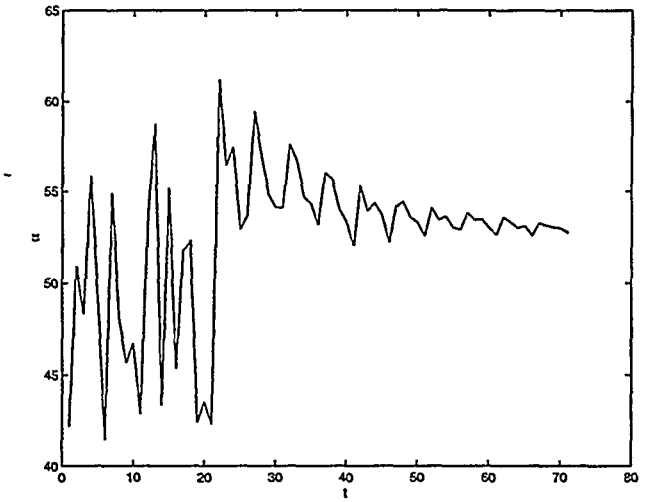

Figure 7: α parameterization in the walk behavior using L_SOM learning.

3.5.2 Acquired Behavioral Sequence

Figures 6(a)-(e) give a series of snapshots on a complete Athlete motion sequence in the given virtual environment. The complete composite motion behaviors are the result of Athlete dual-level behavioral self-organization that starts with blank behavioral maps, and undertakes nearly 200 steps of high-level behavioral conditioning and 1,000 steps of low-level parameterization map updating.

During the course of dual-level behavioral acquisition, the α values are constantly selected and updated. The acquired values may be applied in various high-level pattern conditioning attempts. Figures 7 to 11 provide five typical α self-organization history plots for walk (as acquired from Area 1), hop (Area 6), jump (Area 4), run (Area 1), and squat_walk (Area 2), respectively.

One of the important results should be mentioned here is that once the H_SOM and L_SOMs are built and continuously refined with actual motion data, they can readily be re-used in the subsequent motion behaviors as well as the learning of new ones. Such a reusability is well reflected in the following respect: Whether or not the present inputs, e.g., the external and internal states to the high-level map and the desired motion requirement to the low-level map, exactly match with the previously seen inputs, the agent can always associate them with its best reaction as computed from its earlier conditioning experience. For instance, once Athlete in a standing position, encounters the Flat_Track_Field conditions again in Areas 3, 5, 7, and 9 with a familiar initial velocity, it would immediately adopt a walk behavior for the same motion requirement as has been acquired from Area 1. The same is also true in determining behavioral parameter α.

4 Concluding Remarks

In this paper, we described an implemented and validated approach to generating reactive behaviors in lifelike agents. While presenting the underlying algorithm as well as the specific constructs for dual-level behavioral acquisition, we also provided an example of such an agent, called Athlete, that could interact with its virtual environment and at the same time learn a sequence of coherent parameterized motion behaviors. The presented agent was generated using an implemented software system.

In our presently implemented system, we allowed a lifelike agent to utilize a repository of predefined coarse behavioral pat-

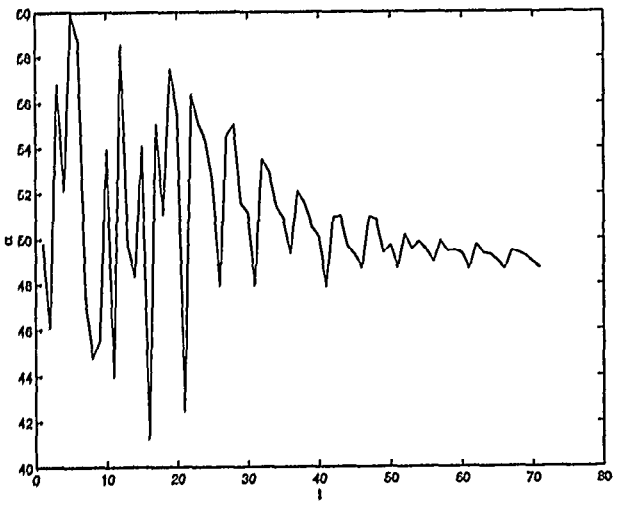

Figure 8: α parameterization in the hop behavior using L_SOM learning.

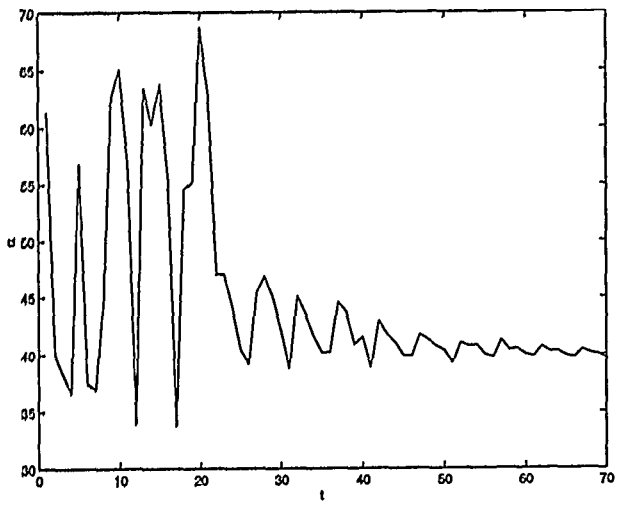

Figure 9: α parameterization in the jump behavior using L_SOM learning.

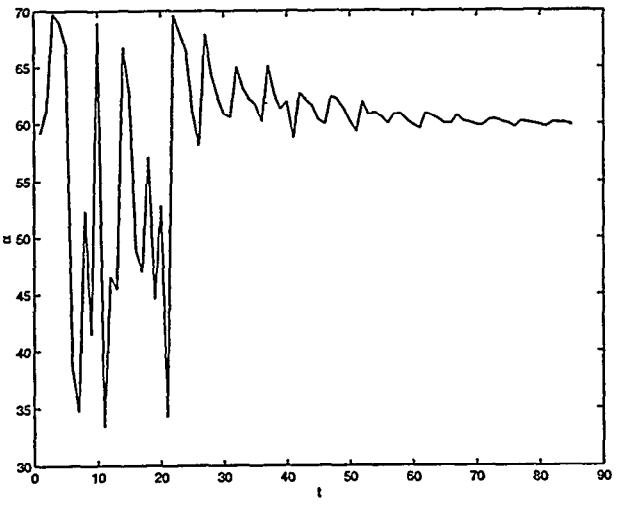

Figure 10: α parameterization in the run behavior using L_SOM learning.

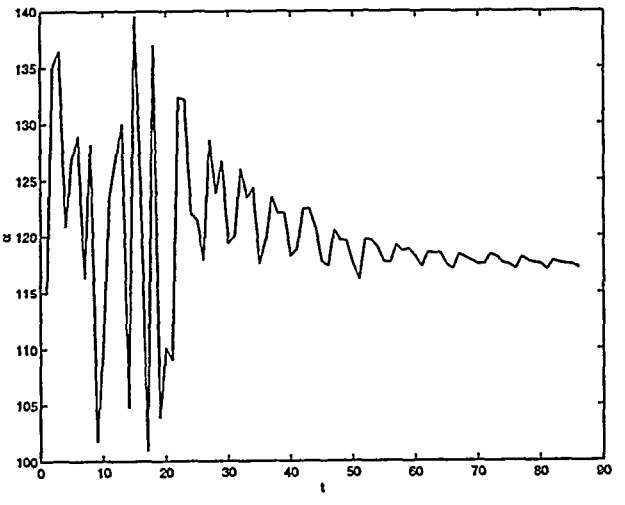

Figure 11: α parameterization in the squat_walk behavior using L_SOM learning.

terns. By doing so, we significantly cut down the computational cost that might otherwise be incurred if the patterns were either kinematically or dynamically synthesised from the first principles.

The objective of our work was to demonstrate an effective way of generating autonomous behaviors in lifelike agents, with practical applications to computer entertainment in mind. Nevertheless, this work could also have a direct bearing on the development of other autonomous agent-based systems that proactively deal with unpredictable, dynamically changing (physical and computational) environments.

As one of the future extensions from our present work, we shall explore the integration of even higher levels of self-organization, called I_SOM and E_SOM, that deal, respectively, with the intention and the emotion of a lifelike agent.

Appendix: Quantitative Relationships in Behavioral Patterns

In what follows, we give the definition for the five primitive behavioral patterns as used in our example. Each of the pattern, i.e., the positional and velocity characterization, is represented in terms of parameter α, as shown in Figure 4, and the initial velocity of the agent. With respect to the links in Figure 2, we assume that L_{11}, L_{12}, L_{21}, and L_{22} all have the same length l.

1. walk:

$$v = V_0 \tag{3}$$

$$\Delta x = 2l\sin(\frac{\alpha}{2}) \tag{4}$$

$$\Delta y = 2l[\cos(\frac{\alpha}{2}) - 1] \tag{5}$$

$$v' = V_0 \tag{6}$$

2. hop:

$$v = V_0 \tag{7}$$

$$\Delta x = l[\cos(\alpha) + \sin(0.81818\alpha - 1.28520)] + 0.86721v\sin^2(\frac{\alpha}{2}) \tag{8}$$

$$\Delta y = l[\cos(0.27273\alpha - 0.42840) + \cos(0.45455\alpha - 0.71400) - 2] \tag{9}$$

$$v' = V_0 \tag{10}$$

3. jump:

$$v = 0.7765V_0 \tag{11}$$

$$\Delta x = l[\cos(\alpha) + \sin(0.8\alpha - 1.25664)] + 0.86721v\sin^2(\frac{\alpha}{2}) \tag{12}$$

$$\Delta y = l[\cos(0.6\alpha - 0.94248) + \cos(1.2\alpha - 1.8850) - 2] \tag{13}$$

$$v' = V_0 \tag{14}$$

4. run:

$$v = [1 + 0.38384\sin^2(\frac{\alpha}{2})]V_0 \tag{15}$$

$$\Delta x = l[\sin(0.41176\alpha) + \sin(0.05882\alpha)] + v\sin^2(\frac{\alpha}{2}) \tag{16}$$

$$\Delta y = l[\cos(0.17647\alpha) + \cos(0.29412\alpha) - 2] \tag{17}$$

$$v' = v \tag{18}$$

5. squat_walk:

$$v = V_0 \tag{19}$$

$$\Delta x = l[\sin(0.35932\alpha) - \sin(0.79167\alpha)] \tag{20}$$

$$\Delta y = l[\cos(0.35932\alpha) + \cos(0.79167\alpha) - 1] \tag{21}$$

$$v' = V_0 \tag{22}$$

where V_0 denotes the initial velocity of the Athlete main body. In the present example, we set $V_0 = 0.59500$ as the initial velocity of the agent at its global starting point in the virtual environment. v and v' denote the velocities of the Athlete main body during and immediately after the motion, respectively. Δx and Δy are the Cartesian displacements of the Athlete main body in $x-$ and $y-$ directions, respectively.

References

[1] Kiyoshi Arai. Keyframe animation of articulated figures using partial dynamics. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Models and Techniques in Computer Animation*, pages 243–256. Springer-Verlag, Tokyo, 1993.

[2] Joel Auslander, Alex Fukunaga, Hadi Partovi, Jon Christensen, Lloyd Hsu, Peter Reiss, Andrew Shuman, Joe Marks, and J. Thomas Ngo. Further experience with controller-based automatic motion synthesis for articulated figures. *ACM Transactions on Graphics*, 14(4):311–336, 1995.

[3] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 37(7), July 1994.

[4] James C. Bezdek and Nikhil R. Pal. A note on self-organizing semantic maps. *IEEE Transactions on Neural Networks*, 6(5):1029–1036, 1995.

[5] Bruce M. Blumberg and Tinsley A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics Proceedings,SIGGRAPH-95, Los Angeles*, Aug. 1995 1995.

[6] Clark Elliott. I picked up Catapia and other stories: A multimodal approach to expressivity for "emotionally intelligent" agents. In *Proceedings of the First International Conference on Autonomous Agents (ACM/AAAI), Marina del Rey, California*, pages 451–457, Feb. 5-8, 1997.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley publishing company, Reading, MA, 1989.

[8] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[9] T. Kohonen. *Self-organization and Associative Memory*. Springer-Verlag, Berlin, 1989.

[10] Jiming Liu, Hong Qin, Y. Y. Tang, and Y. T. Wu. Adaptation and learning in animated creatures. In *Proceedings of the First International Conference on Autonomous Agents (ACM/AAAI), Marina del Rey, California*, pages 371–377, Feb. 5-8, 1997.

[11] Stephania Loizidou and Gordon J. Clapworthy. Legged locomotion using HIDDS. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Models and Techniques in Computer Animation*, pages 257–269. Springer-Verlag, Tokyo, 1993.

[12] Pattie Maes. A bottom-up mechanism for behavior selection in an artificial creature. In J. A. Meyer and S. Wilson, editors, *Proceedings of the first International Conference on Simulation of Adaptive Behavior*. The MIT Press, Cambridge, MA, 1991.

[13] Pattie Maes. Modeling adaptive autonomous agents. *Artificial Life*, 1(1-2), 1994.

[14] Pattie Maes, T. Darrell, B. Blumberg, and A. Pentland. The ALIVES system: Wireless, full-body interaction with autonomous agents. *The ACM Special Issue on Multimedia and Multisensory Virtual Worlds*, Spring 1996.

[15] Karl Sims. Evolving 3D morphology and behavior by competition. In Rodney A. Brooks and Pattie Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 28–39. The MIT Press, Cambridge, MA, 1994.

[16] Demetri Terzopoulos, Xiaoyuan Tu, and Radek Grzeszczuk. Artificial fishes with autonomous locomotion, perception, behavior, and learning in a simulated physical world. In Rodney A. Brooks and Pattie Maes, editors, *Artifical Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 17–27, Cambridge, MA, 1994. The MIT Press.

[17] Nickos Vasilonikolidakis and Gordon J. Clapworthy. Design of realistic gaits for the purpose of animation. In Nadia Magnenat Thalmann and Daniel Thalmann, editors, *Computer Animation '91*, pages 101–114. Springer-Verlag, Tokyo, 1991.

[18] Jeffrey Ventrella. Explorations in the emergence of morphology and locomotion behavior in animated characters. In Rodney A. Brooks and Pattie Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 436–441. The MIT Press, Cambridge, MA, 1994.